



Data Management and Visualization

Politecnico di Torino

Data warehousing in Oracle – Practice 1

The practice purpose is to write some queries, in extended SQL, to retrieve data from the data warehouse described in Point 1. Furthermore, it will be necessary to create some materialized views in order to improve the performance of frequent queries.

1. Problem specifications

A telephone company is interested in analyzing its own data to improve customer services. At present, the company has a database with call logs. For each call, the caller and receiver phone numbers, the duration, the type of charge (e.g., peak, off-peak rates), the start time (date, hour, minute, second) are known. The managers want to obtain very fast the information about the telephone traffic on the company lines and the daily income based on the caller location, the day and the phone rate.

In particular, the managers want to analyze the following situations:

- Monthly net income and number of calls for each caller city.
- Monthly net income and number of calls for each receiver city.
- Monthly net income and number of calls for each caller province and region.
- Monthly net income and number of calls for each receiver province and region.
- Daily net income and number of calls for each caller province.
- Yearly net income and number of calls for each caller province and region.
- Monthly net income and number of calls for each phone rate (type of charge).
- Net income and number of calls for each day of the week and phone rate.
- Daily number of calls for each caller region.
- Daily number of calls for each receiver region.

The OLTP database of the telephone company is reported in Table 1.

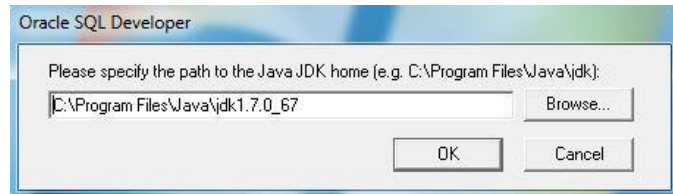
2. Querying the data warehouse

In Table 2 the proposed solution for the data warehouse is shown. The corresponding tables have already been created in Oracle and they contain some sample data. Use these tables as source for the following queries.

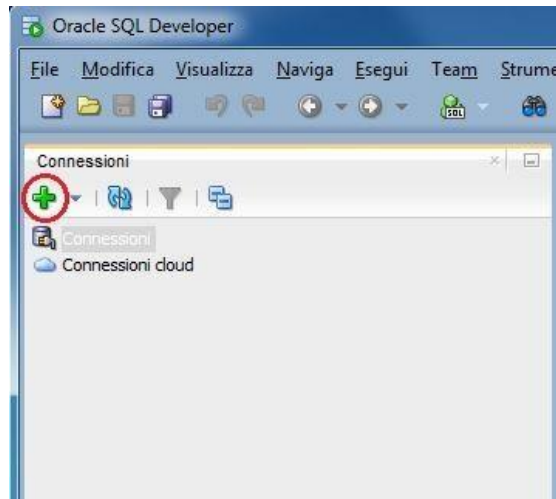
1. Select the yearly income for each phone rate, the total income for each phone rate, the total yearly income and the total income.
 2. Select the monthly number of calls and the monthly income. Associate the RANK() to each month according to its income (1 for the month with the highest income, 2 for the second, etc., the last month is the one with the least income).
 3. For each month in 2003, select the total number of calls. Associate the RANK() to each month according to its total number of calls (1 for the month with the highest number of calls, 2 for the second, etc., the last month is the one with the least number of calls).
 4. For each day in July 2003, select the total income and the average income over the last 3 days.
 5. Select the monthly income and the cumulative monthly income from the beginning of the year.
- Estimate the cardinality of the data warehouse tables (facts and dimensions) and decide whether and which materialized views are needed to improve performance of the previous queries.
 - Create the materialized views you consider convenient and compare the execution plan cost of the queries using and without using the materialized views.

3. Connection to the database

1. Open the Oracle SQL Developer program (from Start Menu-All programs)
2. Select the Java SDK path



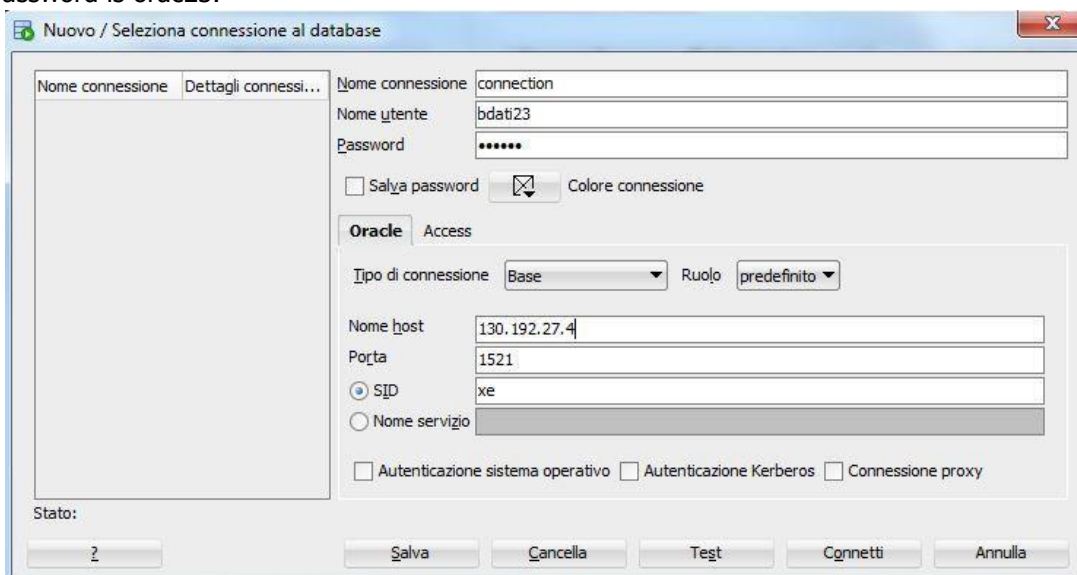
3. Click on the green "plus" button on the left to create a new connection



4. Login

- a. To logon you have to insert the following parameters:
 - i. Nome utente (username): bdati[choose a number between 1-100]
 - ii. Password: orac[choose a number between 1-100]
 - iii. Nome host (host name): 130.192.27.4
 - iv. Port: 1521
 - v. SID: xe

For example, if you are working on pc number 23, the corresponding username is bdati23 and the password is orac23.



5. Execute the query and create the materialized views by means of the interface.

Tables	Description
DWABD . PHONERATES (phoneRateType INT NOT NULL, phoneRateName VARCHAR(20) NOT NULL, phoneRate_CostPerSecond FLOAT NOT NULL, PRIMARY KEY(phoneRateType));	Different phone rates 7 rows
DWABD . PLACES (Places_ID INT NOT NULL, City VARCHAR(20) NOT NULL, Province VARCHAR(20) NOT NULL, Region VARCHAR(20) NOT NULL, PRIMARY KEY(Places_ID));	Places 1500 rows
DWABD . CALLS (CallerPhoneNumber VARCHAR(20) NOT NULL, ReceiverPhoneNumber VARCHAR(20) NOT NULL, CallerLocation INT NOT NULL, ReceiverLocation INT NOT NULL, FullDate DATE NOT NULL, StartTimeHour INT NOT NULL, StartTimeMinute INT NOT NULL, StartTimeSecond INT NOT NULL, CallDuration FLOAT NOT NULL, phoneRateType INT NOT NULL, PRIMARY KEY(CallerPhoneNumber,ReceiverPhoneNumber,FullDate,StartTimeHour ,StartTimeMinute,StartTimeSecond), FOREIGN KEY(phoneRateType) REFERENCES PhoneRates(phoneRateType) ON DELETE CASCADE, FOREIGN KEY(CallerLocation)REFERENCES Places(Places_ID) ON DELETE CASCADE, FOREIGN KEY(ReceiverLocation) REFERENCES Places(Places_ID) ON DELETE CASCADE);	Calls in 2003 and 2004 ~ 1300000 rows

Table 1 – Source data base with single call information

Tables	Description
DWABD . TIMEDIM <pre>(ID_time INT NOT NULL, DayOfWeek CHAR(15) NOT NULL, DateMonth INT NOT NULL, DateYear INT NOT NULL, PRIMARY KEY(ID_time));</pre>	Time dimension 10 rows
DWABD . PHONERATE <pre>(ID_phoneRate INTEGER NOT NULL, phoneRateType VARCHAR(20) NOT NULL, PRIMARY KEY(ID_phoneRate));</pre>	Phone rate dimension 7 rows
DWABD . LOCATION <pre>(ID_location INTEGER NOT NULL, City VARCHAR(20) NOT NULL, Province CHAR(20) NOT NULL, Region CHAR(20) NOT NULL, PRIMARY KEY(ID_location));</pre>	Place dimension 1500 rows
DWABD . FACTS <pre>(ID_time INTEGER NOT NULL, ID_phoneRate INTEGER NOT NULL, ID_location_Caller INTEGER NOT NULL, ID_location_Receiver INTEGER NOT NULL, Price FLOAT NOT NULL, NumberOfCalls INTEGER NOT NULL, PRIMARY KEY(ID_time, ID_phoneRate, ID_location_Caller, ID_location_Receiver), FOREIGN KEY(ID_time) REFERENCES timeDim(ID_time), FOREIGN KEY(ID_phoneRate) REFERENCES phoneRate(ID_phoneRate), FOREIGN KEY(ID_location_Caller) REFERENCES location(ID_location), FOREIGN KEY(ID_location_Receiver) REFERENCES location(ID_location));</pre>	Fact table 7809 rows

Table 2 – Proposed solution - Data warehouse tables